I.J. Information Technology and Computer Science, **2017, 3, 10-18**
Published Online March 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2017.03.02

ATAM-based Architecture Evaluation Using LOTOS Formal Method

Muhammad Usman Ashraf
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: m.usmanashraf@yahoo.com

Wajdi Aljedaibi
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: waljedaibi@kau.edu.sa

***Abstract*—**System Architecture evaluation and formal specification are the significant processes and practical endeavors in all domains. Many methods and formal descriptive techniques have been proposed to make a comprehensive analysis and formal representation of a system architecture. This paper consists of two main parts, in first we evaluated system performance, quality attribute in Remote Temperature Sensor clients-Server architecture by implementing an ATAM model, which provides a comprehensive support for evaluation of architecture designs by considering design quality attributes and how they can be represented in the architecture. In the second part, we computed the selected system architecture in ISO standards formal description technique LOTOS with which a system can be specified by the temporal relation between interactions and behavior of the system. Our proposed approach improves on factors such as ambiguity, inconsistency and incompleteness in current system architecture.

***Index Terms*—**ATAM, Architecture, LOTOS, CADP, Software Quality Attributes Evaluation, Software Quality Assurance.

I. Introduction

Architecture assessment has become more important due to the ever-increasing complexities in software and system development. Indeed, system architectural analysis at early stages detects and removes maximum flaws with minimum effort and cost [18]. In addition, accurate selection of system architecture is vital for time to market of critical systems [19]. Many approaches have been proposed to evaluate the architectural designs of a system at early stage, including Scenario-based Architecture Analysis (SAAM), Performance Assessment of Software Architecture (PASA), Architecture Level Modifiability Analysis (ALMA) etc. [20]. Nevertheless, early stage evaluation of large-scale system architecture is not sufficiently addressed by existing architectural evaluation methods.

There is a tradeoff in designs vs. discipline that play a role to correctly direct efforts at such initial level of a system. We selected a Remote Temperature Sensor (RTS) [4] client server architecture as an example to implement the Architecture Tradeoff Analysis Method (ATAM), which provides support to discover dependencies among elements and quality attributes of an architecture design at early stages of product lifecycle [21]. Further, we executed the selected architecture in the Language of Temporal Ordering Specification (LOTOS) [9], which is an ISO standard formal description technique from which an initial prototype can be generated to get immediate feedback from the client on the basis of elicited requirements [23].

The reminder of the paper is organized as follows: a background on ATAM and LOTOS is given in section 2. Section 3 describes Remote Temperature Sensor client server architecture (RTS) and its functionality. ATAM evaluation of selected RTS architectures is given in section 4. Section 5 presents the LOTOS specification of ATAM architectures for RTS. We also show RTS architecture graphically as generated by CADP [11]; a toolset to compile and execute LOTOS specifications.

II. Background

In this section we introduce the basic overview of software architecture, architecture tradeoff analysis method (ATAM) and its phases and steps in detail by implementation on real life scenarios.

A. Software Architecture

Every program has an architecture which is comprised of different pieces/ components that interact in a deterministic way. Similarly in a software system, software architecture is, basically, a structured set of an interactive elements, which constitute different software parts, their visible properties and interconnection between them. A well designed architecture is a complete description of how the system elements interact with each other. An intensive software system is described as a static software structure on which design time elements depend on, controls dynamic software structure runtime elements and the interconnection between them [1]. With respect to visible properties of software architecture elements, a system is noticed by its behavior (what the system will do) and its quality properties (how the system will do it) such as availability, scalability, security, ubiquity, usability etc. However, a software architecture could be designed in different ways based on software requirements and desired quality attributes.

According to one principle of architecture "Every computer program has an architecture, whether or not it is documented and understood" [24]. The architecture of a system is a fundamental property that demonstrates whether or not it has been documented and understood. The right architecture paves the way of the system, whereas a wrong architecture usually spells some form of disaster. A software architecture designing is very early and important stage where an architect has to take most decisions on how the development should proceed. Once an architecture of a system is built, it is hard to change in later, therefore all the decisions should be in the right way under the limitation of the requirements and quality attributes of the system as well [1].

B. ATAM

System design evaluation is a key analytical process in all disciplines and intellectual and practical endeavors. In terms of software architecture design analysis, many methods have been proposed in order to analyze a system architecture such as Scenario-based Architecture Analysis (SAAM), Performance Assessment of Software Architecture (PASA), Architecture Level Modifiability Analysis (ALMA) and Architecture Tradeoff Analysis Method (ATAM). ATAM provides a comprehensive support for evaluation of architecture designs since it allows consideration of multiple quality attributes such as reliability, portability, performance, usability, security, etc. The origins of ATAM started with software architecture analysis method (SAAM) with ambitions to consider most common quality attributes such as modifiability and performance [4]. There is a tradeoff that must be considered, i.e. some qualities may conflict such dependability vs performance. As a result, a tradeoff method is highly required at initial levels of the system development. ATAM tradeoff analysis helps to discover the dependencies among the elements and attributes of an architecture.

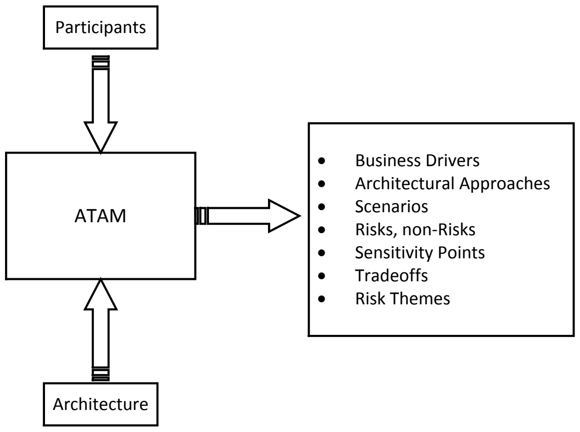

Fig.1. ATAM Input Output [6]

ATAM extracts the architecture of a system as well as stakeholder participation and business goals to emphasize the attention of the evaluators on the portion of the architecture that is central to the achievement of the goals. ATAM takes some business drivers and architectural documents as input and produce a valuable output with involvement of some participants of the method as described in below fig.1 [5]. ATAM gets an input as an architecture and produces the output of some participant of the method. Output from ATAM is utilized to prepare a final written report.

C. ATAM Phases and Steps

ATAM method activities are categorized into four phases as shown in fig.2 and explained below:

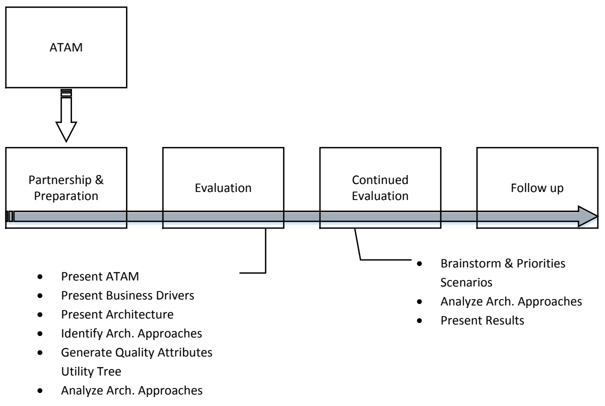

Fig.2. ATAM Phases and Steps [6]

- ***Phase "0" Partnership and Preparation***

The first phase of ATAM is "partnership and preparation" takes a little over few weeks under the leadership of the evaluation team and project decision makers. It involves a set of informal meetings between them to sort out the work in detail.

- ***Phase "1" Evaluation***

The second phase of ATAM is the evaluation of the system at initial level. The evaluation team evaluates the system architecture, its functional and non-functional requirements as detailed in the requirements document. This phase takes normally couple of days at which a formal meeting is arranged between project decision maker and evaluation team to gather the information and analysis. This phase consists of several steps where ATAM method is presented by the evaluation leader to the project representatives. In the next step, all project representatives are involved in evaluating and understating the primary business drivers and overall system's business perspectives [12]. In step 3 of this phase, a detailed architecture of the system at an appropriate level is presented by an architect where all technical limitations are addressed such as OS, hardware, and software limitations are presented and discussed. An important decision at this step is the selection of architectural approaches that are suitable for system requirements. In step 5, evaluation team and project decision makers articulate the quality attributes in detail and prioritize them according to the system requirements. These quality attributes are discussed in the form of different scenarios. In the final step, the architect explains the architectural support against these scenarios.

- ***Phase "2" Evaluation (Continued)***

This phase has three steps which may last around 2-3 days to complete where project decision makers, evaluation team, and stakeholders review all what has been learnt from phase 1 and summarize them. In this phase, more scenarios could be considered and analysed. The methodology presents to stakeholder what is going to be implemented for this particular system and share all the risks, non-risks, trade-off, and sensitive factors. Moreover, in order to brainstorm scenarios, evaluation team discusses with stakeholder which scenarios are more meaningful with respect to the stakeholders' individual roles and then order them accordingly. Once the scenarios are finalized, evaluation team presents the high ranked and newly, if added, scenarios to the architect. In last step of this phase, the collected information is documented and presented to stakeholder.

- ***Phase "3" Follow-Up***

The last phase of ATAM is related to self-examination of architecture evaluation team where they discuss the advantages, disadvantages, obstacles, and all decisions related to the architecture devised. They go through all surveys and conducted meetings in phase 0, 1, 2 to generate a final report from the scenarios.

D. Formal Description Approach

Due to the ever increasing complexities of technology and its advancement in many facets of our surrounding environment, system reliability and correctness have become major concerns during software development projects. An appropriate address to such concerns lies within formal system specification methods that embodies mathematical rigor and precision to verify system properties [8]. The use of formal specification approach increases confidence on quality factors such as reliability, performance, availability, ambiguity, inconsistency and incompleteness in current system architecture.

The Language of Temporal Ordering Specification (LOTOS) is a formal description language developed by International standard organization (ISO) for open system formal specification. Systems in LOTOS are specified by drawing the temporal relation between interactions establishing the discernible behaviour of the system [3]. LOTOS is one of the Formal Description Techniques (FDTs) built on precise mathematical semantics that can be implemented in different architectural styles and approaches [2]. Specifying a system in LOTOS describes both the static and dynamic behaviours of the system. Particular properties are described in different ways in different methods which lead to problem in other methods to ensure after developing the system model the descriptions remain invariant [2]. In addition, LOTOS specifications, due to its strong mathematical basis, can be executed at an early stage in the development project generating an initial level prototype from which immediate client feedback can be gathered.

LOTOS syntax can be written in two different styles depending on the desired level of abstraction as set forth by architect. Basic LOTOS is an abstract style in which an architect can specify the basic interactions and synchronizations between concurrent processes, while full LOTOS is a more expressive style that allow for additional language operators to accommodate for complex conditions, parameters, and return values. LOTOS code can be compiled by CAESAR compiler after which it can be used in many ways. For example, CADP allows for running the specification enabling the architect to experiment the architecture in a live mode and examine flow of events. Furthermore, the compiler generates a Binary Coded Graph (BCG) file which is a formatted labelled transition system. BCG file is further compiled using BCG_DRAW tool to generate the final graphical representation of the architecture [9].

III. RTS CLIENTS-SERVER ARCHITECTURE

In order to elaborate a system architecture in ATAM, we used a common Remote Temperature Sensor (RTS) system which used to measure the temperature of all the furnaces placed in it, full details of RTS can be found in [4]. The basic principle of RTS is that an operator, might be host computer, sends specific frequency to RTS to get the furnace temperature according to specified frequency. Once RTS receives the frequency from host, it gets the temperature from a set furnaces in analog form and forwards it to Analog to Digital Converter (ADC) which converts it into digital form. Temperatures are then placed in a queue because ADC can convert only one furnace temperature at a time. By following frequency restrictions, ADC converts the temperature from analog to digital form and reports to system operator.

In this section we illustrated RTS (Remote Temperature Sensor) an example architecture analyzed using ATAM method. RTS system could be used in different architectures such as clients-Server architecture, two servers multiple clients and Client-Intelligent Cache Server architecture. RTS architecture selection depends on system requirement and quality attributes for that particular system. For instance, system requirement may enforce to implement clients-Server architecture with limited quality attributes that provide furnace report to any client. In contrast, if a quality attribute such as "availability" is a requirement for our RTS system that should be implemented with limited cost constraints, then rebuilding architecture by using two servers and multiple-clients architecture is a candidate architectural model. However, if the system requires additional quality attributes, for example "performance", a client-intelligent cache server model maybe considered at which an extra wrapper, i.e. intelligent cache, can analyze variation in furnace temperature by its cool down or heat up levels [10].

In this paper we selected the client-server architectural model for our RTS example, as shown in fig.3. The selected architecture consists of single server and eight clients as shown in fig.3-a. A detailed description of RTS server that contains ADC and similar number of furnaces as clients is shown in fig.3-b.

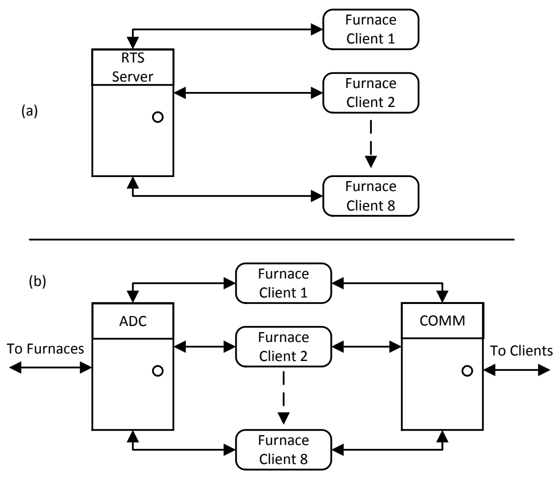

Fig.3. RTS Client-Server Architecture

IV. ATAM Implementation in Clients-Server Architecture

In this section we show the phase activates when applying ATAM on our selected RTS architecture, which are as follows:

- Partnership and Preparation (Phase '0')
- Present ATAM (Phase '1' Step 1) already discussed above.
- Present Architecture (Phase '1' Step 3) already discussed above.
- Analyze Architectural Approach (Phase 1, Step 6)

These ATAM activates were selected for this particular RTS architecture; ATAM provides many steps from which a suitable subset can be chosen. Next, two more ATAM activates are needed for evaluation of the architecture:

- RTS requirements and constraints within a client-server architecture can be viewed as:
  - Receive client's request.
  - Supply furnace temperature under specific frequency constraint.
  - Provide temperature periodic update to client at particular rate.
- RTS architecture & attribute utility tree, which is evaluated and explained in details in the next section.

A. *RTS Architecture & Performance Quality Attributes*

RTS system requires a prompt response within specific frequency rate. However, according to above business drivers and constrains the high priority quality attribute is asserting system performance. Other quality attributes, such as availability, security etc., might be required, however, in this paper we only consider performance quality. Real-time RTS system architecture frequently implicate with jitter and latency that are important for furnace temperature report. Latency for system performance could be analysed by depicting execution paths [13] [14]. A latency determination model consists of several factors including paths and its related execution time. The following parameters are of concern as follows:

- Temperature transmission time over the network C_{net}
- Queuing time of Input / output for temperature report C_{dq}
- ADC: Analog to Digital converter
- ADC Processing time and periodic reporting C_{fnc}

Using these parameters, latency can be computed by finding Best Case Periodic Latency (BCPL) And Worst Case Periodic Latency (WCCL). The best case could be occurred when the queue (Q) requests from clients to server is zero as Q = 0. In contrast, worst case scenario occurs when all clients (C) associated to server (S) control request for (F) furnace temperatures simultaneously as shown in Table 1. WCPL and BCPL can be defined as [4]:

$$WCPL = C/S \times F \times (C_{net} + C_{dq} + C_{fnc}) \quad (1)$$

$$BCPL = (C_{net} + C_{dq} + C_{fnc}) \quad (2)$$

Table 1. RTS Client Server Performance Summary

WCPL (Sec)	BCPL (Sec)	PL_{HL} (Sec)	PL_{ML} (Sec)	$Jitter_{HL}$ (Sec)	$Jitter_{ML}$ (Sec)
12.16	0.19	12.16	12.16	11.97	589

In case of significant variation in control request, processing and periodic report, and time equation will be changed to these control requests and periodic report. Generally, when at same time all clients associated to server are scheduled for furnace reading, latency of periodic report will be as in equation 3.

$$Q = C/S \times F$$

$$PL = (Q + 1) \times (C_{net} + C_{dq} + C_{fnc}) \quad (3)$$

In order to determine jitter which is the variation in latency from the BCPL or ideal case, (4) can be applied:

$$Jitter = PL - BCPL \quad (4)$$

For instance, in term of performance evaluation, C = 8 number of clients request to S server for temperature of F = 8 number of furnaces. We assume that the network transmission time C_{net} for this scenario is 100 ms (milliseconds) and operation time C_{dq} for queue is 10 ms. The request processing time C_{fnc} can be adjusted as 80ms; then according to (1) worst case periodic latency is as follows:

$$WCPL = 8 / 1 \times 8 \times (100 + 10 + 80) => 64 \times (190)$$
$$WCPL = 12160$$

Where

$$BCPL = (100 + 10 + 80) \quad => 190$$

Periodic latency in moderate and heavy load cases:

$$PL_{HL} = (64) \times (100 + 10 + 80) \quad => 64 \times 190$$
$$PL_{HL} = 12160$$
$$PL_{HL} = WCPL$$

Where, for moderate case, the number of clients can be minimized to 4 furnaces.

$$PL_{ML} = 4 / 1 \times 8 \times (100 + 10 + 80) => 32 \times 190$$
$$PL_{ML} = 6080$$

Similarly, the worst and best cases of jitter can be computed by applying (4) & (5):

$$Jitter_{HL} = 12160 - 190 = 11970$$
$$Jitter_{ML} = 6080 - 190 = 5890$$

B. RTS Architecture Quality Attributes Tree

Table 2. RTS Architecture Quality attributes and Scenarios

Quality Attribute	**Attribute Refinement**	**Scenarios**
Performance	Temperature Response Time	Server respond periodic temperature report to clients at specific rate received from Furnaces.
	Furnace temperature response frequency	Furnace respond to Server according to frequency sent by server.
	Analog to Digital Temperature Conversion processing	ADC receives analog temperature from furnace, convert it into digital form and send to operator.
	Throughput	At peak level, System is able to respond at .19 per second with Best Case Periodic Latency.

Since RTS scenarios are associated with only the performance quality attribute, the required quality attribute is refined and presented explicitly according to generated scenarios in ATAM step 2.

V. LOTOS FORMAL MODEL

In order to specify System architecture in LOTOS, we have used CADP: an interactive tool for construction and analysis of distributed processes [11]. CADP is formally known as "CAESAR/ALDEBARAN Development Package" that provides an extensive number of functionalities for design and analysis of multiple architectures and processes. It was developed by CAVCS team to provide support for several languages compilation and specification [11], and offers the tools such as:

- Compiler for various patterns.
- Various verification algorithms.
- Different model checkers for several temporal logics
- Supportive for different Equivalence checker tools.
- Performance Evaluations.
- Simulation and evaluation.
- Visual validation and verification.

CADP provides a systematic way for LOTOS specification and compilation to generate desired output. The CADP life cycle which is required to generate an output of LOTOS code in graphical form is shown in fig. 4. In phase 1, the desired architecture is specified by following ATAM principles and written in LOTOS syntax. All ATAM scenarios should be present in the code so that the architecture can be fully and correctly exercised. Compiling the LOTOS specification in CADP produces the BCG, which is required for further handling of the specifications. Once the BCG file is generated, CADP uses it to reason about many aspects of the specification including running it and/or showing a graphical representation of the specification, hence a visual representation of the architecture. In phase 3, CADP toolset takes in the BCG file as its input to produce a graphical representation of the architecture which is the final step as shown in fig.4.

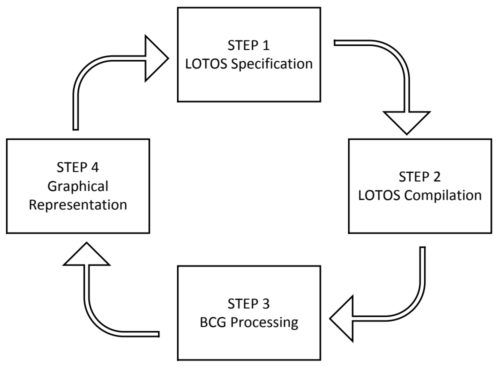

Fig.4. CADP steps for handling LOTOS Specification

A. RTS Architecture Specification in LOTOS

In this section we present the LOTOS specification code of RTS client server architecture using basic LOTOS. RTS System architecture specification consist of two parts as behaviour and Processes.

1. Behaviour part contains process synchronization at an abstract level using hiding operator. In order to achieve parallel composition we use interleaving operator (|||) among different processes [16, 17].
2. Process part consist of all processes where each process has its name, list of interaction points, behaviour expression, and list of parameters.

```
specification RTS_Client_Server [Req_Temp, Prov_Temp] :
noexit behaviour
hide Temp_Freq, A_Temp, D_Temp, Temp_update,
Convert_Temp, Recieve_Temp in
(
  (
  Client [Req_Temp, Temp_Freq,Prov_Temp, Recieve_Temp]
  |||
  Server [Req_Temp,Prov_Temp, Temp_Freq, Temp_update]
  |||
  communicator [Req_Temp,Prov_Temp]
  )
 |[Temp_Freq, A_Temp, D_Temp, Temp_update,
Convert_Temp, Recieve_Temp]|
 (
 RTS_Furnace [Temp_Freq,Temp_update, A_Temp ,D_Temp]
 |||
 ADC [A_Temp,D_Temp,Convert_Temp]
 )
)
where
process
Server[Req_Temp,Prov_Temp,Temp_Freq,Temp_update]:
noexit:=....endproc (*Server Process*)
process
Client[Req_Temp,Temp_Freq,Prov_Temp,Recieve_Temp]:
noexit:=...endspec(*Clients Process*)
process
RTS_Furnace[Temp_Freq,Temp_update,A_Temp,D_Temp]:
noexit:=....endproc(*Furnace Process*)
process ADC [A_Temp,D_Temp,Convert_Temp] :
 noexit := .... endproc  (*ADC Process*)
process communicator [Req_Temp,Prov_Temp]:
noexit := .... endproc  (*Communicator Process*)
```

Listing 1: RTS Architecture LOTOS Specification

Listing 1 shows the LOTOS code for RTS architectural modules, where language keyword are noted in bold. In line 1, ***specification*** *RTS_Client_Server* synchronizes with environment through the two formal gates *Req_Temp* and *Prov_Temp* [15]. For instance, when a client requests temperature reading of specific furnace at particular frequency, there is an internal interaction between furnaces and the server, which should be hidden from client side. However, the internal functionalities and actions of processes are abstracted using the ***hide*** operator in line 2. Lines 3 to 10 show the parallel composition between clients, server, and communicator model between them. As our model consists of multiple clients and furnaces, their processes are specified generically to enable reuse and hence accomplish possible system scalability. A generic form of client process is described as listing 2.

```
process Client [Req_Temp, Temp_Freq, Prov_Temp,
   Recieve_Temp] : noexit :=
   Req_Temp;    (* request to server for temprature *)
   Server[Req_Temp,Prov_Temp, Temp_Freq,Temp_update]
>> Recieve_Temp;
   []
   Prov_Temp;exit
Endspec
```

Listing 2: RTS Clients Process Module LOTOS Specification

In listing 2 *Client* process consist of four formal gates parameters: *Req_Temp, Temp_Freq, Prov_Temp,* and *Recieve_Temp*. A ***noexit*** in first line implies that *Client* must recursively perform an operation either

1. Send request to server for any furnace temperature, *or*
2. Receive temperature from server side.

However, after receive the temperature reading, the Client is forced to exit. We assume that clients get activated by another process each time a temperature is requested. A choice operator '[]' to used to allow for selection since a Client can have one of two behaviors: request temperature (Req_Temp), or receive temperature (Prov_Temp.) For example, when requesting a temperature reading from server, it calls server process along required parameters as temp_freq and 'Receive_temp' to receive furnace temperature. Similarly at server process side, shown in listing 3, *Server* process can either provide temperature to clients or send temperature frequency to *RTS_Furnace* and call furnace process for further processing.

```
process Server [Req_Temp,Prov_Temp, Temp_Freq,Temp_update]:
      noexit :=
Req_Temp;       (* if client ask for temprature *)
      >> Prov_Temp;
      Client [Req_Temp, Temp_Freq,Prov_Temp, Recieve_Temp]
[]
Temp_Freq;
      RTS_Furnace [Temp_Freq,Temp_update, A_Temp,D_Temp]
[]
>> Temp_update;exit
endproc
```

Listing 3: RTS Server Process Module LOTOS Specification

RTS_Furnace process, shown in listing 4, gets the temperatures at specific frequency and forwards them to ADC in analog form. ADC process, shown in listing 5, has three formal gates:

1. *A_Temp* for analog temperature received from furnaces,
2. *D_Temp* for digital temperature that returns back after conversion, and
3. *Convert_Temp* a device that converts the temperature.

ADC gets a single request from queue and converts it into digital form. According to ADC process, there is always A_Temp in first step to produces D_Temp. Once, A_Temp is received, it creates a call to Convert_Temp for temperature conversion and then processes D_Temp value. Finally, it returns furnace temperature in digital form which is further forwarded to *Client* according to required frequency rate.

```
process        RTS_Furnace[Temp_Freq,Temp_update,A_Temp,
    D_Temp] : noexit :=
  Temp_update OR Temp_Freq; (* Get Request of Temperature
    update from Server *)
   (
   ADC [A_Temp,D_Temp,Convert_Temp];
    (* Provide Analog temperature to ADC and request for Digital
    temperature update *)
   )
   >> D_Temp;
   Server[Req_Temp,Prov_Temp,Temp_Freq, Temp_
          update]
exit
endproc
```

Listing 4: RTS Furnace Process Module LOTOS Specification

```
process ADC [A_Temp,D_Temp,Convert_Temp] :
  noexit :=
  A_Temp;
          (* Get Analog temprature *)
    (
    Convert_Temp;
  (* Convert temprature *)
    )
    >> D_Temp;
    RTS_Furnace [Temp_Freq,Temp_update,
  A_Temp,D_Temp] exit
endproc
```

Listing 5: RTS ADC Process Module LOTOS Specification

VI. Results

Producing an executable architecture in LOTOS is shown to be a promising step towards establishing confidence that an architecture satisfies initial requirements and constraints. Running the architecture in LOTOS is an additional plus that can help to develop the correct system. Our proposed RTS architecture was initially devised and evaluated in ATAM, then specified in LOTOS using CADP tool to produce a running version of the architecture as shown in fig5. A *Client* process sends a temperature reading request *Req_Temp* to *Server* process. However, we notice that there is an intermediatory process *communicator*, which is needed to facilitate communication between possibly differing *Client –Server* platforms. Once the *Server* receives the request, it forwards it as a *Temp_Freq* message to all furnace processes *RTS_Furnace* for temperature reading. A conversion process must be completed before the returning the reading to the Server, mainly because our furnaces produce analog based temperature readings. For this reason, *RTS_Furnace* first forwards the temperature in a *Temp_update* message to the *ADC* process, which performs analog-to-digital conversion, then passes the digital temperature reading as *D_Temp* back to *RTS_Furnce* processes. At this stage, a value returning process starts when *RTS_Furnace* returns the temperature reading *Temp_Update* in digital format to the *Server* process, which will ultimately returns the readings to *Client* processes.

Before generating this computable architecture through CADP tool, a manual architecture was drawn which was specified in LOTOS. It should be noted at such an early stage of development of a system, all formal gates should be assigned meaningful titles to allow for proper and easy understanding of the architecture specification. In our LOTOS specification we implemented eight clients that interact with a single RTS server to get temperatures from different furnaces. However, results clearly shows that the selected architecture is accurate and performed well.

VII. Conclusion

Architecture evaluation and formal description are needed to see if certain requirements and quality attributes can be realized gracefully at the final software product. Accurate architecture selection at early stages of development is crucial for time to market of critical systems. In order enhance the system verification and validation according to given requirement, we introduced an approach to evaluate software architecture and performance quality attribute of an RTS Clients-Server Architecture. According to our approach, the first encouraging result is that we applied current software architecture evaluation methods and showed how a required quality attributes can be expressed in ATAM. We then measured the performance of the selected architecture and showed how to calculate its functionality in a way that signifies one or more quality attribute, e.g.

performance. Furthermore, we presented client-server based RTS architecture in LOTOS formal description technique and generated graphical representation by executing through CADP tools. Our work show that the proposed approach for evaluating and formally presenting architectures in LOTOS is useful for proper architecture selection. This approach is applicable to evaluate and validate the requirements, constraints, and quality attributes of any system.

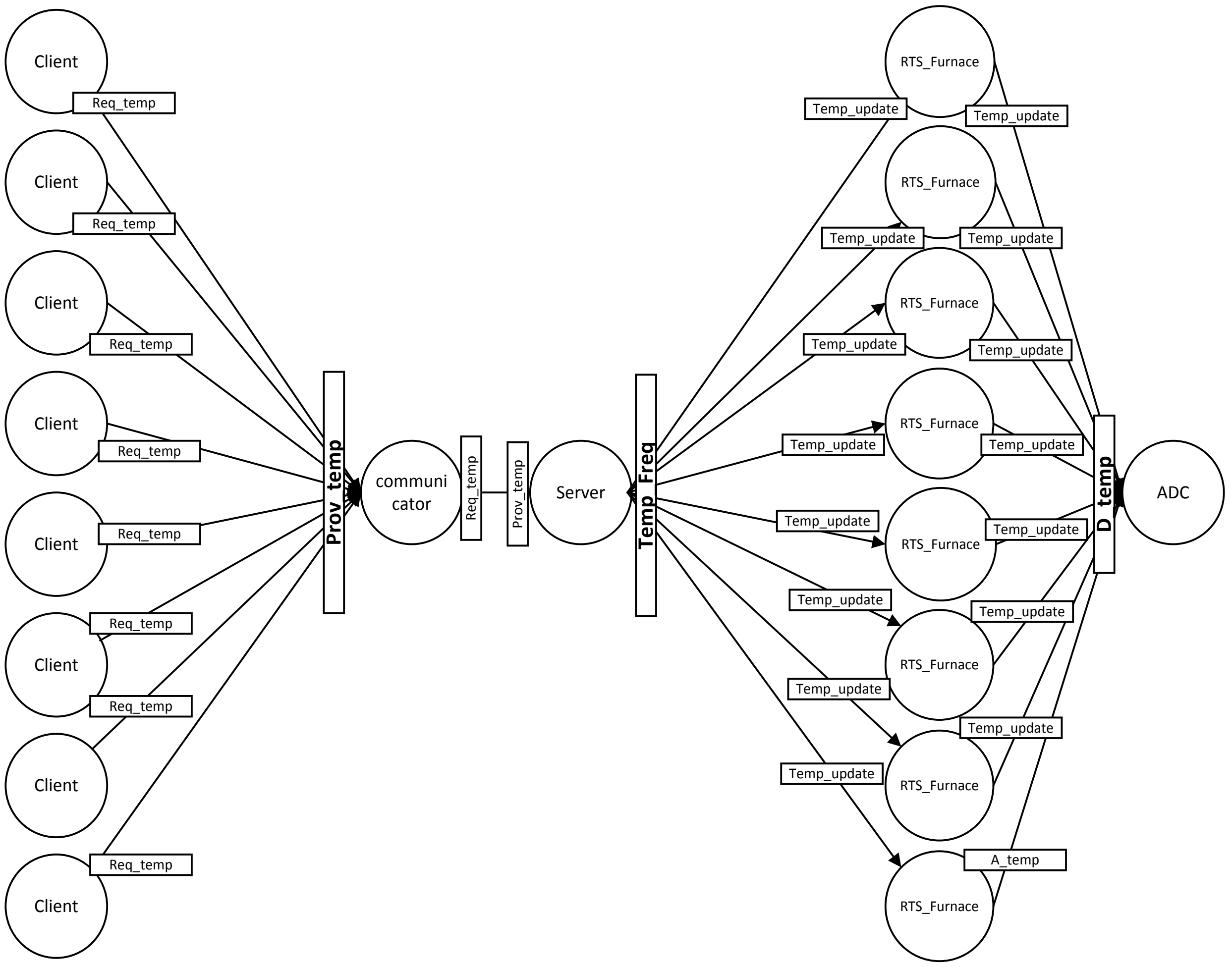

Fig.5. CADP output of LOTOS Specification

References

[1] Shaw, Mary, and David Garlan. *Software architecture: perspectives on an emerging discipline*. Vol. 1. Englewood Cliffs: Prentice Hall, 1996.

[2] Moreira, Ana MD, and Robert G. Clark. "Combining object-oriented analysis and formal description techniques." *Object-Oriented Programming*. Springer Berlin Heidelberg, 1994. 344-364.

[3] Bolognesi, Tommaso, and Ed Brinksma. "Introduction to the ISO specification language LOTOS." *Computer Networks and ISDN systems* 14.1 (1987): 25-59.

[4] Kazman, Rick, et al. "The architecture tradeoff analysis method." *Engineering of Complex Computer Systems, 1998. ICECCS'98. Proceedings. Fourth IEEE International Conference on*. IEEE, 1998.

[5] Nord, Robert L., et al. *Integrating the Architecture Tradeoff Analysis Method (ATAM) with the cost benefit analysis method (CBAM)*. No. CMU/SEI-2003-TN-038. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2003.

[6] Bass, Len. *Software architecture in practice*. Pearson Education India, 2007.

[7] Lopez, Marta. *An evaluation theory perspective of the Architecture Tradeoff Analysis Method (ATAM)*. No. CMU/SEI-2000-TR-012. carnegie-mellon univ pittsburgh pa software engineering inst, 2000.

[8] Clarke, Edmund M., and Jeannette M. Wing. "Formal methods: State of the art and future directions." *ACM Computing Surveys (CSUR)* 28.4 (1996): 626-643.

[9] Lotos, I. S. O. "A formal description technique based on the temporal ordering of observational behaviour." *International Organisation for Standardization-Information Processing Systems-Open Systems Interconnection, Geneva* (1988).

[10] Kazman, Rick, et al. "The architecture tradeoff analysis method."*Engineering of Complex Computer Systems, 1998. ICECCS'98. Proceedings. Fourth IEEE International Conference on*. IEEE, 1998.

[11] Garavel, Hubert, et al. "CADP 2006: A toolbox for the construction and analysis of distributed processes." *International Conference on Computer Aided Verification*. Springer Berlin Heidelberg, 2007.

[12] Aladwani, Adel M. "Online banking: a field study of drivers, development challenges, and expectations." *International Journal of Information Management* 21.3 (2001): 213-225.

[13] Audsley, N. C. et al. “Fixed Priority Pre-Emptive Scheduling: An Historical Perspective.” Real-Time Systems 8, 2-3 (March-May 1995): 173-198.

[14] Conway, R.; Maxwell, W.; & Miller, L. Theory of Scheduling. Reading, MA: Addison-Wesley Publishing Company, 1967.

[15] Logrippo, Luigi, Mohammed Faci, and Mazen Haj-Hussein. "An introduction to LOTOS: learning by examples." *Computer Networks and ISDN systems* 23.5 (1992): 325-342.

[16] Poizat, Pascal, Christine Choppy, and Jean-Claude Royer. "Concurrency and data types: A specification method an example with LOTOS." *International Workshop on Algebraic Development Techniques*. Springer Berlin Heidelberg, 1998.

[17] Ardis, Mark A. "Lessons from using basic lotos." *Proceedings of the 16th international conference on Software engineering*. IEEE Computer Society Press, 1994.

[18] Zalewski, Andrzej, and Szymon Kijas. "Beyond ATAM: Early architecture evaluation method for large-scale distributed systems." *Journal of Systems and Software* 86.3 (2013): 683-697.

[19] Rupanov, V., et al. "Employing early model-based safety evaluation to iteratively derive E/E architecture design." *Science of Computer Programming* 90 (2014): 161-179.

[20] Dobrica, Liliana, and Eila Niemelä. "A survey on software architecture analysis methods." *Software Engineering, IEEE Transactions on* 28.7 (2002): 638-653.

[21] Närman, Per, et al. "Enterprise architecture availability analysis using fault trees and stakeholder interviews." *Enterprise Information Systems* 8.1 (2014): 1-25.

[22] Moreira, Ana MD, and Robert G. Clark. "Combining object-oriented analysis and formal description techniques." *Object-Oriented Programming*. Springer Berlin Heidelberg, 1994. 344-364.

[23] El-Gendy, Hazem, Nabil El Kadhi, and Narayan Debnath. "Towards sound development of PIXITP, conformance test suites, and conforming implementations for various Formal Description Techniques." *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*. IEEE, 2008.

[24] Clements, Paul, et al. "The duties, skills, and knowledge of software architects." *2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*. 2007.

Authors' Profiles

Usman M. Ashraf received his B.Sc. degree from Govt. College Gujranwala in 2007, M.Sc. degree in Computer Science from The University of Agriculture Faisalabad in 2009 and Master of Science in Computer Science from University of Lahore, Pakistan in 2014. Currently, he is doing Ph.D. in computer science from King Abdul-Aziz Saudi Arabia. His research interests include Exascale computing System, High Performance Computing, Ubiquitous Computing and Context awareness. He has presented many papers in National and International conferences.

Wajdi Aljedaibi is a faculty member of the Computer Science department in the Faculty of Computing & Information Technology (FCIT) at King Abdulaziz University. He served as the KAU IT Manager and then Dean of Information Technology at KAU. Wajdi was awarded the Ph.D. in Information Technology – Software Engineering and MSc. in Software Systems Engineering both from George Mason University. His current research interests are: CMMI-based evaluations and methods, component-based software engineering, software measurement, and ERP system.

How to cite this paper: Muhammad Usman Ashraf, Wajdi Aljedaibi,"ATAM-based Architecture Evaluation Using LOTOS Formal Method", International Journal of Information Technology and Computer Science (IJITCS), Vol.9, No.3, pp.10-18, 2017. DOI: 10.5815/ijitcs.2017.03.02